JOBSHEET

BASIS DATA LANJUT

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG

TAHUN AJARAN 2025/2026

PERTEMUAN 2

Query Dasar (DDL dan DML) pada PostgreSQL

Team Teaching Basis Data Lanjut:

- Candra Bella Vista, S.Kom., MT.
- Moch Zawaruddin Abdullah, S.ST., M.Kom.
- Yan Watequlis Syaifudin, ST., MMT., PhD.
- Yoppy Yunhasnawa, S.ST., M.Sc.

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah : Basis Data Lanjut
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 3 (tiga)
Pertemuan ke- : 2

JOBSHEET 02

Query Dasar (DDL dan DML) pada PostgreSQL

1. Data Definition Language (DDL)

DDL atau *Data Definition Language* adalah bagian dari SQL yang digunakan untuk mendefinisikan struktur dan obyek basis data, seperti membuat, mengubah, atau menghapus tabel, indeks, dan view. DDL tidak berhubungan langsung dengan manipulasi data, tetapi lebih fokus pada pengelolaan objek database itu sendiri. Perintah yang digunakan dalam DDL meliputi:

Klausa	Deskripsi
CREATE	Perintah untuk membuat objek
ALTER	Perintah untuk melakukan perubahan struktur/objek, seperti tabel atau kolom
DROP	Perintah untuk menghapus objek, seperti database, table, view, index

1.1. CREATE

Merupakan perintah DDL yang digunakan untuk mendefinisikan obyek database, seperti database, tabel, indeks, dan view. Berikut adalah sintaks untuk membuat database:

```
CREATE DATABASE namadatabase;
```

Nama database maupun nama tabel tidak boleh mengandung spasi. Nama database juga tidak boleh sama antar database dalam satu server/cluster. Selain membuat database, perintah CREATE juga digunakan untuk mendefinisikan tabel. Berikut adalah sintaks untuk membuat tabel:

```
CREATE TABLE namatabel(
      Namakolom1 tipedata constraint,
```

```
        Namakolom2 tipedata,
        .....
);
```

Nama tabel tidak boleh sama dalam satu skema. Nama kolom tidak boleh sama dalam satu tabel, namun boleh sama di tabel lain. Pembuatan indeks dan view akan dijelaskan pada pertemuan selanjutnya.

1.2. ALTER

Setelah membuat tabel dalam database, dapat memodifikasi field pada tabel yang telah dibuat. Perintah ALTER erupakan perintah DDL yang digunakan untuk mengubah struktur tabel. Berikut ini adalah beberapa query ALTER:

a. Menambahkan constraint

```
ALTER TABLE namatabel ADD CONSTRAINT namakolom_pk
PRIMARY KEY(namakolom);
```

b. Menambahkan kolom

```
ALTER TABLE namatabel ADD COLUMN namakolom tipedata;
```

c. Menghapus kolom

```
ALTER TABLE namatabel DROP COLUMN namakolom;
```

d. Mengubah tipe data

```
ALTER TABLE namatabel ALTER COLUMN namakolom TYPE
tipedatabaru;
```

e. Mengubah nama kolom

```
ALTER TABLE namatabel RENAME COLUMN namakolom tipedata;
```

1.3. DROP

Perintah DROP adalah bagian dari DDL yang digunakan untuk menghapus obyek basis data. Berikut adalah sintaks untuk perintah DROP:

a. Menghapus tabel

```
DROP TABLE namatabel;
```

b. DROP TABLE …. CASCADE

Perintah ini memungkinkan PostgreSQL untuk menghapus semua obyek yang bergantung pada tabel tersebut. Misalnya: FOREIGN KEY di tabel lain yang mereferensikan tabel tersebut, view/materialized view yang memakai tabel tersebut.

```
DROP TABLE namatabel CASCADE;
```

c. Menghapus database

```
DROP DATABASE namadatabase;
```

2. Data Manipulation Language (DML)

Data Manipulation Language adalah bagian dari SQL yang digunakan untuk memanipulasi data dalam tabel database. DML mencakup operasi-operasi untuk menambah (*create*), mengambil data (*retrieve/read*), mengubah (*update*), dan menghapus data (*delete*) yang ada di dalam database atau yang sering disebut dengan operasi CRUD.

Perintah-perintah dalam DML digunakan untuk bekerja dengan data yang telah ada di dalam tabel, berbeda dengan DDL (Data Definition Language) yang digunakan untuk mendefinisikan atau mengubah struktur objek dalam basis data.

2.1. INSERT

Perintah INSERT digunakan untuk menambahkan satu atau lebih baris data ke dalam tabel. Terdapat dua cara untuk melakukan query INSERT, yaitu menambahkan satu per satu data baru pada tabel atau menambahkan beberapa baris sekaligus. Berikut adalah sintaks untuk single row insert dan multi row insert:

a. *Single Row Insert* → Menambahkan satu data baru pada tabel

```
INSERT INTO namatabel (namakolom1, namakolom2, ...)
VALUES (datakolom1, datakolom2, ...);
```

b. *Multi Row Insert* → Menambahkan beberapa baris data baru sekaligus pada tabel

```
INSERT INTO namatabel(namakolom1, namakolom2, ...)
VALUES (datakolom1, datakolom2, ...),
       (datakolom1, datakolom2, ...),
       (datakolom1, datakolom2, ...);
```

2.2. SELECT

Perintah SELECT digunakan untuk mengambil data dari satu atau lebih tabel dalam database. Khusus perintah SELECT ini, bisa dimasukkan pada kategori *Data Retrieval Language* (DRL). Perintah untuk menampilkan semua data dari satu tabel:

```
SELECT * FROM namatabel;
```

Perintah untuk menampilkan data berdasarkan kondisi tertentu:

```
SELECT * FROM namatabel WHERE kondisi;
```

2.3. UPDATE

Perintah UPDATE digunakan untuk mengubah data yang sudah ada di dalam tabel. Perintah ini memerlukan klausa WHERE untuk memfilter baris data yang ingin diubah. Karena jika tanpa klausa WHERE, maka semua data dalam tabel akan ikut terganti, dan hal ini tentu saja tidak baik. Berikut adalah sintaks perintah UPDATE:

```
UPDATE  namatabel
SET     namakolom = databaru
WHERE   kondisi;
```

2.4. DELETE

Perintah DELETE digunakan untuk menghapus data dari tabel berdasarkan kondisi yang diberikan dalam klausa WHERE. Tanpa klausa WHERE, perintah ini akan menghapus seluruh data dalam tabel dan hal ini tentu saja tidak baik. Berikut adalah contoh sintaks query DELETE:

a. Menghapus data berdasarkan satu kondisi tertentu

```
DELETE FROM namakolom
WHERE  kondisi;
```

b. Query yang Tidak Disarankan (tidak ada kondisi WHERE)

```
DELETE FROM namakolom;
```

Menggunakan query DELETE tanpa ada kondisi WHERE tidak disarankan, karena query akan bekerja lebih lambat dan memicu row-level trigger. Jika ingin mengosongkan isi tabel tetapi tetap menjaga struktur tabel, gunakan perintah TRUNCATE.

2.5. TRUNCATE

Perintah TRUNCATE digunakan untuk menghapus seluruh data dalam tabel, namun struktur tabel tetap ada. Ini lebih cepat daripada DELETE karena tidak memerlukan pencatatan transaksi untuk setiap baris yang dihapus. Berikut adalah sintaks untuk perintah TRUNCATE

```
TRUNCATE TABLE namatabel1, namatabel2, … opsitruncate
```

PostgreSQL mendukung beberapa opsi untuk perintah TRUNCATE, yaitu:

a. RESTART IDENTITY: reset sequence yang dimiliki tabel ke awal
b. CONTINUE IDENTITY (default): tidak mereset sequence
c. CASCADE: ikut truncating tabel lain yang mereferensikan tabel via FK
d. RESTRICT: mencegah jika ada tabel lain yang mereferensikan

3. Tipe Data PostgreSQL

Tipe data adalah spesifikasi jenis nilai yang boleh disimpan beserta operasi yang valid atas nilai tersebut. Pada basis data, tipe data menentukan domain kolom, cara penyimpanan, akurasi, pemakaian indeks, dan menjaga integritas data (misalnya: mencegah teks masuk ke kolom tanggal). PostgreSQL mendukung tipe data secara umum yang dimiliki oleh DBMS, diantaranya:

Tipe Data	Keterangan
INT	Angka bulat
FLOAT, DOUBLE	Angka pecahan
DATE	Tanggal YYYY-MM-DD
DATETIME	Tanggal dan waktu YYYY-MM-DD HH-MM-SS
DECIMAL(p,s)/ NUMERIC(p,s)	Angka presisi tetap
CHAR, VARCHAR, TEXT	Data teks
BOOLEAN	Boolean
BLOB	Biner

Selain tipe data yang disebutkan pada tabel di atas, terdapat beberapa tipe data khas PostgreSQL yang tidak dimiliki oleh MySQL, yaitu:

Tipe Data	Keterangan
SERIAL	Auto increment integer
UUID	Universally Unique Identifier
JSONB	JavaScript Object Notation, tetapi dalam format biner terkompresi
ARRAY	Satu kolom dapat menyimpan banyak nilai sekaligus
MONEY	Menyimpan nilai mata uang
RANGE	Menyimpan rentang nilai

3.1 SERIAL

Serial merupakan tipe data khusus yang dimiliki oleh PostgreSQL. Jika suatu kolom memiliki tipe data serial, setiap kali melakukan insert data, PostgreSQL akan otomatis memberikan nilai unik yang bertambah (1,2,3,….). Berikut adalah beberapa variasi tipe data serial:

Variasi	Rentang nilai
SMALLSERIAL	1 – 32.767
SERIAL	1 – 2.147.438.674 (INT 32-bit)
BIGSERIAL	1 – 9.223.372.036.854.775.807 (INT 64-bit)

3.2 UUID

UUID atau Universally Unique Indentifier, merupakan jenis tipe data khas PostgreSQL. Tipe data ini adalah nilai yang berupa string 128-bit yang hampir mustahil duplikat, walaupun dibuat di komputer/server yang berbeda. Format tipe data ini berupa 36 karakter terdiri dari 32 bilangan hexa decimal (0-9) dan (a-f), serta 4 tanda strip. Contohnya: 550e8400-e29b-41d4-a716-446655440000.

3.3 JSONB

Tipe data PostgreSQl untuk menyimpan data JSON (JavaScript Object Notation), tetapi dalam format biner terkompresi. Dibandingkan dengan tipe JSON biasa, JSONB lebih cepat untuk query dan indexing. Berikut adalah perbandingan JSON vs JSONB:

Aspek	JSON	JSONB
Penyimpanan	Menyimpan teks mentah JSON	Menyimpan biner terstruktur
Kecepatan	Lebih lambat	Lebih cepat, bisa pakai indeks
Urutan Key	Dipertahankan	Tidak dipertahankan
Duplikasi Key	Bisa disimpan	Dihilangkan

3.4 ARRAY

Tipe data PostgreSQl yang memungkinkan satu kolom menyimpan banyak nilai sekaligus, mirip seperti array dalam bahasa pemrograman. Contoh: kolom 'keahlian' bisa meyimpan ['AI', 'Data Mining", 'Basis Data'] dalam satu baris. Array digunakan untuk data yang memang berupa list singkat, tetapi jika data akan sering dianalisis (join/filter kompleks) sebaiknya tetap dibuat tabel relasi M:N, bukan array.

3.5 MONEY

Tipe data PostgreSQl untuk menyimpan nilai mata uang. Nilainya otomatis ditampilkan dengan symbol mata uang sesuai pengaturan bahasa/region server PostgreSQL. Dapat dilakukan perhitungan (SUM, +, -), tetapi untuk operasi yang lebih kompleks kurang fleksibel, lebih baik dikonversi ke dalam format NUMERIC.

3.6 RANGE

Tipe data PostgreSQl untuk menyimpan rentang nilai. Bisa digunakan untuk angka, tanggal, waktu. PostgreSQL menyediakan operator khusus untuk mengecek apakah suatu nilai masuk dalam range. Format dasar:

[ ] inklusif (termasuk batas),
( ) ekslusif (tidak termasuk batas)

Contohnya:

[1, 5] → 1 sampai 5, termasuk 1 dan 5
[1, 5) → 1 sampai 5, termasuk 1, tidak termasuk 5

Berikut adalah tipe range built-in:

Tipe	Isi rentang
Int4range	Rentang INTEGER
Int8range	Rentang BIGINT
Numrange	Rentang NUMERIC
Daterange	Rentang DATE
Tsrange	Rentang TIMESTAMP tanpa zona
Tstzrange	Rentang TIMESTAMP dengan zona

4. Constraint

Constraint merupakan batasan atau aturan yang ada pada tabel. PostgreSQL menyediakan beberapa tipe constaint berikut:

a. NOT NULL merupakan suatu kolom yang mendifinisikan dengan constraint NOT NULL. Kolom yang berfungsi sebagai kunci primer (Primary Key) otomatis tidak boleh NULL
b. UNIQUE mendifinisikan suatu kolom bersifat unik, artinya antara satu data dengan data lain namanya tidak boleh sama, misal alamat email.
c. PRIMARY KEY Constaint PRIMARY KEY membentuk key yang unik untuk suatu tabel.
d. FOREIGN KEY Constraint didefinisikan pada suatu kolom yang ada pada suatu tabel, dimana kolom tersebut juga dimiliki oleh tabel yang lain sebagai suatu PRIMARY KEY bisa digunakan untuk menghubungkan antara dua tabel.
e. CHECK constraint yang satu ini mendefinisikan sebuah kondisi untuk data agar dapat masuk dalam field artinya tiap pemasukan data atau editing terhadap data record, field yang dimasukkan akan selalu diperiksa apakah isinya ada diantara data-data yang dimasukkan, jika tidak ada maka SQL akan menampilkan pesan ERROR.
f. EXCLUDE hanya dimiliki PostgreSQL, melarang kombinasi yang "bertabrakan" menurut operator tertentu (memerlukan indeks GIST/SP-GIST). Misal: jadwal: satu ruang tidak boleh punya rentang waktu yang beririsan.

5. PostgreSQL vs MySQL

Secara umum sintaks DDL dan DML yang dimiliki oleh PostgreSQL mirip dengan MySQL. Pada pendefinisian tabel menggunakan PostgreSQL, terdapat perbedaan tipe data yang digunakan. Misalnya pada kolom ID, pada MySQL fungsi auto increment menggunakan constraint AUTO_INCREMENT, sedangkan pada PostgreSQL digunakan tipe data SERIAL/BIGSERIAL untuk melakukan auto increment pada kolom tertentu. PostgreSQL mendukung perintah DROP TABLE … CASCADE, yang artinya dapat menghapus tabel meskipun tabel direferensikan oleh FOREIGN KEY tabel lain. Pada MySQL jika ada tabel direferensikan oleh FOREIGN KEY, maka FOREIGN KEY harus dihapus terlebih dahulu constrainnya, atau DROP tabel dependent (anak) terlebih dahulu.

Pendefinisian sintaks DML pada PostgreSQL dan MySQL juga mirip. Perbedaannya adalah pada perintah TRUNCATE. Baik PostgreSQL maupun MySQL sama-sama mendukung perintah TRUNCATE. Bedanya TRUNCATE pada MySQL otomatis mereset sequence auto-increment, sedangkan pada PosgreSQL secara default akan melanjutkan sequence auto-increment, tetapi memiliki opsi RESTART IDENTITY. PostgreSQL mendukung TRUNCATE … CASCADE, tetapi MySQL tidak. TRUNCATE pada MySQL tidak dapat di rollback, tetapi

pada PostgreSQL bisa di rollback dalam transaksi. Tabel berikut merangkum perbandingan DDL dan DML pada PostgreSQL vs MySQL:

Aspek	MySQL	PostgreSQL
AUTO INCREMENT	AUTO_INCREMENT	Tipe data SERIAL
DROP TABLE ... CASCADE	Tidak ada	Ada
ON DELETE / UPDATE CASCADE	Ada	Ada
TRUNCATE … CASCADE	Tidak ada	Ada
Constraint EXCLUDE	Tidak ada	Ada

Praktikum 01 – Data Definition Language

1. Buka DBeaver
2. Buatlah sebuah database dengan nama perpustakaan_namamasing-masing

```
create database perpustakaan_xyz;
```

3. Buatlah tabel buku dengan struktur sebagai berikut:

Field	Tipe Data	Constraint
ID	UUID	PRIMARY KEY
Judul	VARCHAR(50)	NOT NULL
Tahun_terbit	INT	CHECK, tahun_terbit antara 2000 – tahun sekarang

```
CREATE TABLE buku (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  judul VARCHAR(50) NOT NULL,
  tahun_terbit INT CHECK (tahun_terbit BETWEEN 2000 AND EXTRACT(YEAR FROM CURRENT_DATE)::INT)
);
```

4. Buatlah tabel anggota dengan struktur sebagai berikut:

Field	Tipe Data	Constraint
ID	SERIAL	PRIMARY KEY
Nama	VARCHAR(30)	NOT NULL
Email	VARCHAR(30)	
Created_at	TIMESTAMPTZ	NOT NULL, default sekarang

```sql
CREATE TABLE anggota (
  id          SERIAL PRIMARY KEY,
  nim         VARCHAR(30),
  nama        VARCHAR(30) NOT NULL,
  email       VARCHAR(30) UNIQUE,
  created_at  TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

5. Buat tabel petugas dengan struktur sebagai berikut (soal 1):

Field	Tipe Data	Constraint
ID	SERIAL	PRIMARY KEY
Nama	VARCHAR(30)	NOT NULL
Email	VARCHAR(30)	

6. Buat tabel kategori dengan struktur sebagai berikut (soal 2):

Field	Tipe Data	Constraint
ID	SERIAL	PRIMARY KEY
Nama_kategori	VARCHAR(30)	NOT NULL

7. Tambahkan kolom dengan aturan berikut pada tabel buku (soal 3):

Field	Tipe Data	Constraint
Penulis	ARRAY TEXT (TEXT[])	NOT NULL
Pengarang	VARCHAR(20)	NOT NULL
Kategori_id	INT	FK ID Tabel Kategori

8. Buat tabel peminjaman dengan struktur sebagai berikut (soal 4):

Field	Tipe Data	Constraint
ID	SERIAL	PRIMARY KEY
Anggota_id	INT	FK ID Tabel Anggota
Petugas_id	INT	FK ID Tabel Petugas
Tanggal_pinjam	DATE	NOT NULL, default tanggal hari ini
Jatuh_tempo	DATE	NOT NULL

9. Buat tabel detil_peminjaman dengan struktur sebagai berikut (soal 5):

Field	Tipe Data	Constraint
Id	SERIAL	PRIMARY KEY
Peminjaman_id	INT	FK ID Tabel Peminjaman
Buku_id	UUID	FK ID Tabel Buku
Tanggal_kembali	DATE	
Denda	NUMERIC(12,2)	

Praktikum 02 – Data Manipulation Language

1. Tambahkan data pada tabel kategori dengan query berikut:

```
INSERT INTO kategori (nama) VALUES ('Teknologi'), ('Fiksi'), ('Manajemen');
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 6)!

2. Tambahkan data pada tabel anggota dengan query berikut:

```
INSERT INTO anggota (nama, email) VALUES
('Citra Lestari', 'citra@example.ac.id'),
('Budi Santoso',  'budi@example.ac.id'),
('Adi Nugroho', 'adi@example.ac.id');
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 7):!

3. Tambahkan nama dan email anda, serta 3 orang teman anda, kemudian tunjukkan hasilnya (soal 8)!

4. Tambahkan data pada tabel petugas dengan query berikut ini:

```
INSERT INTO petugas (nama, email) VALUES
('Admin Perpus', 'admin.perpus@example.ac.id');
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 9)!

5. Tambahkan data pada tabel buku dengan query berikut:

```
INSERT INTO buku (judul, tahun_terbit, penerbit, penulis, kategori_id)
values ('Dasar Basis Data', 2023, 'Polinema Press', array['Bella, Zawa, Yan, Yoppy'], 1),
       ('Dasar Pemrograman', 2024, 'Polinema Press', array['Vivin, Triana, Vivi'], 1),
       ('Pengantar Manajemen', 2002, 'Penamuda Media', array['Bejo'], 3);
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 10)!

6. Tambahkan data pada tabel buku dengan query berikut:

```
INSERT INTO buku (judul, tahun_terbit, penerbit, penulis, kategori_id)
values ('Harry Potter: The Sorcerers Stone', 1997, 'Bloomsbury', array['JK Rowling'], 2);
```

Jelaskan apa yang terjadi setelah query dieksekusi, dan bagaimana solusinya (soal 11)?

7. Tambahkan data peminjaman dengan query berikut:

```
INSERT INTO peminjaman (anggota_id, petugas_id, tanggal_pinjam, jatuh_tempo)
VALUES (
  (SELECT id FROM anggota WHERE email='citra@example.ac.id' LIMIT 1),
  (SELECT id FROM petugas LIMIT 1),
  CURRENT_DATE,
  CURRENT_DATE + INTERVAL '7 days'
);
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 12)!

8. Tambahkan data detil_peminjaman dengan query berikut:

```
INSERT INTO detil_peminjaman (peminjaman_id, buku_id)
SELECT
  currval(pg_get_serial_sequence('peminjaman','id')) AS peminjaman_id,
  b.id
FROM buku b
WHERE b.judul = 'Dasar Basis Data';
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 12)!

9. Update data detil_peminjaman dengan query berikut:

```
UPDATE detil_peminjaman SET tanggal_kembali = '2025-09-01' WHERE id = 1;
```

Jelaskan apa yang terjadi setelah query dieksekusi (soal 13)!

Tugas Praktikum

Buatlah basis data Toko berdasarkan tabel universal berikut ini:

Kode	Nama	Email	SKU	Nama produk	Harga	Stok	No faktur	Tanggal	Qty
C001	Rani	rani@example.com	CF01	Coffe Latte	25000	50	F001	2025-08-29	2

C001	Rani	rani@example.com	FD02	Croissant	18000	30	F001	2025-08-29	1
C002	Bima	bima@example.com	CF02	Americano	20000	60	F002	2025-08-30	1
C002	Bima	bima@example.com	FD01	Sandwich Tuna	30000	40	F002	2025-08-30	2
C003	Sinta	sinta@example.com	CF01	Coffe Latte	25000	50	F003	2025-08-30	1
C004	Lion	lion@example.com	CF01	Coffe Latte	25000	50	F004	2025-08-30	1
C005	Oni	oni@example.com	FD01	Sandwich Tuna	30000	40	F005	2025-08-31	1

1. Deskripsikan struktur data dari tabel-tabel yang mungkin terbentuk dari tabel universal tersebut!
2. Buatlah tabel-tabel sesuai dengan deskripsi yang telah dibuat pada langkah 1!
3. Tambahkan tabel kategori produk yang terdiri dari kolom id dan nama_kategori. Isikan data seperti berikut:

ID	Nama_kategori
1	Makanan
2	Minuman

4. Relasikan tabel kategori dengan tabel yang sudah dibuat sebelumnya!
5. Ubah stok produk dengan SKU FD01 menjadi 5!
6. Ubah harga pada produk yang memiliki SKU FD01 menjadi 28000!
7. Ubah tanggal transaksi no faktur F005 menjadi tanggal hari ini (tanggal kelas praktikum)!
8. Tampilkan semua data yang ada pada setiap tabel!

**** Sekian, dan selamat belajar ****